UCRL-ID-147309

LLNL Site-Specific ASCI Software Quality Engineering Recommended Practices *Overview*

Version 1.0
February 2002

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Authors

Tim Peck	ASCI Technical Publication Staff
Debra Sparkman	ASCI Verification & Validation Staff
Nancy Storch	ASCI Verification & Validation Staff

Acknowledgments

The following individuals supported the writing of this document, provided guidance and served as reviewers: Lee Busby, Bob Corey, Bob Cooper, Shawn Dawson, Evi Dube, Bruce Hill, Dick Lewis, Steve Louis, Joyce Moulden, Rob Neely, Carolyn Owens, Ed Russell, David Sam, and Nancy Storch.

Revision History

Date	Initials	Comments
26-Nov-2001	drs	Initial Creation
12-Dec-2001	drs	Draft V0.3 Released for Review
23-Jan-2002	drs	Draft V0.4 Released for Review. Grammar and formatting changes included. Added paragraphs to explain the purpose of the Overview document and Table 1.
25-Feb-2002	drs	Draft V1.0 Released for Review. Text converted to outline format.

Table of Contents

Purpose of Recommended Practices

The LLNL Site-Specific Advanced Simulation and Computing (ASCI) Software Quality Engineering Recommended Practices V1.1 document describes a set of recommended software quality engineering (SQE) practices for ASCI code projects at Lawrence Livermore National Laboratory (LLNL). In this context, SQE is defined as the process of building quality into software products by applying the appropriate guiding principles and management practices.

Continual code improvement and ongoing process improvement are expected benefits. Certain practices are recommended, although projects may select the specific activities they wish to improve, and the appropriate time lines for such actions. Additionally, projects can rely on the guidance of this document when generating ASCI Verification and Validation (V&V) deliverables. ASCI program managers will gather information about their software engineering practices and improvement. This information can be shared to leverage the best SQE practices among development organizations. It will further be used to ensure the currency and vitality of the recommended practices.

This *Overview* is intended to provide basic information to the LLNL ASCI software management and development staff from the *LLNL Site-Specific ASCI Software Quality Engineering Recommended Practices V1.1* document. Additionally the Overview provides steps to using the *LLNL Site-Specific ASCI Software Quality Engineering Recommended Practices V1.1* document. For definitions of terminology and acronyms, refer to the Glossary and Acronyms sections in the *LLNL Site-Specific ASCI Software Quality Engineering Recommended Practices V1.1.*

Recommended Practices as a Guide

The LLNL Site-Specific Advanced Simulation and Computing (ASCI) Software Quality Engineering Recommended Practices V1.1, or "Recommended Practices" as the document is commonly designated

- Provides guidance for determining what software development practices and formality of the implementation are appropriate for the risk associated with the software
- Provides assistance if improvements to existing practices are desired

In addition to a section on Guiding Principles and Management Practices, the Recommended Practices document identifies three areas:

- Software Verification
- Software Engineering
- Software Project Management

These practices are related to the *ASCI Software Quality Engineering, Goals, Principles and Guidelines*, frequently referred to as the "Tri-Lab Guidelines." Figure 1 below describes the relationship between the ASCI Tri-Lab Guidelines and the LLNL Recommended Practices.

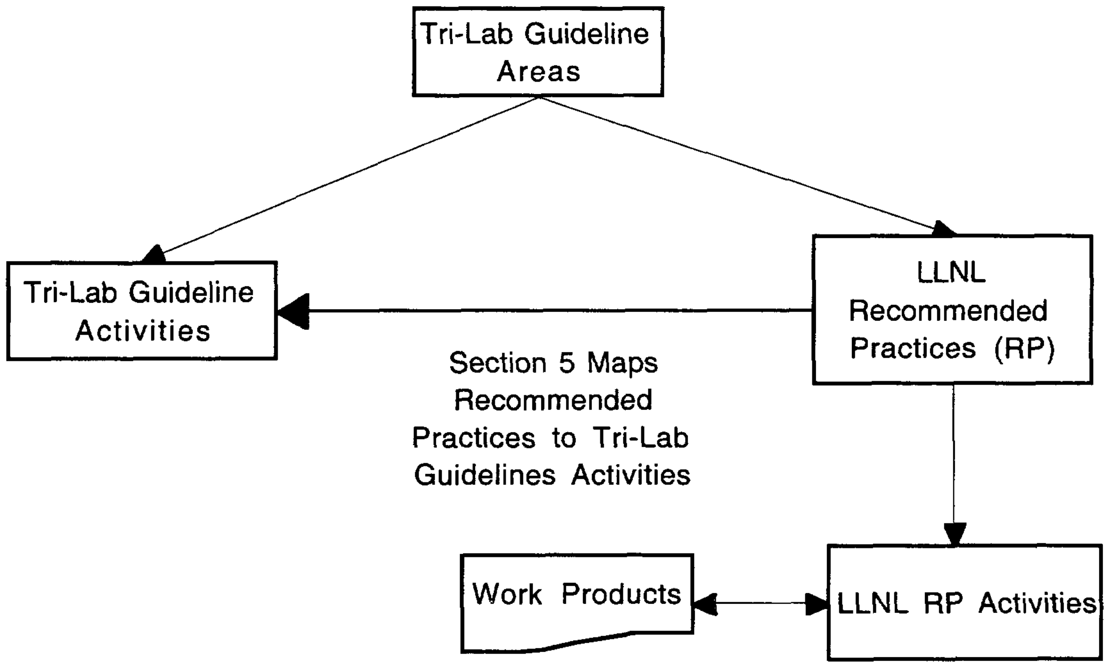
Tri-Lab Guideline Areas
Tri-Lab Guideline Activities
LLNL Recommended Practices (RP)
Section 5 Maps Recommended Practices to Tri-Lab Guidelines Activities
Work Products
LLNL RP Activities

Figure 1. ASCI Tri-Lab Guidelines relationship to LLNL Recommended Practices (RP)

The Recommended Practices (RP) Section 5 provides a table that maps each LLNL RP to a corresponding Tri-Lab Guideline Activity.

- Table is useful to determine how a code team is implementing the Tri-Lab Guidelines
- Each LLNL Recommended Practice contains one or more Activities
- RP Activities in turn produce work products that may be resources for other RP Activities

Using the Recommended Practices

The Recommended Practices document is useful in performing the following five steps:

1. Determining the risk level for a software project
2. Assessing the existing software practices
3. Mapping the results of the existing software practices assessment to the graded approach recommendations
4. Assessing and planning software process improvement (SPI) activities to align with the graded approach recommendations
5. Implementing the software process improvement action plan

Step 1. Determining Risk Level for a Software Project (Section 1.4)

A graded approach to software engineering means that the SQE practices are subjected to a degree of formality commensurate with the following:

- Risk of failure of the software
- Cause of a health or safety issue
- Concern about reliability

Three factors drive the Recommended Practices:

- The selection of specific SQE practices
- The rigor of their implementation
- The strategies for their improvement

Determining the Risk Level

- Initially determine the risk level for the applicable project using the Recommended Practices Table 1.1, Example of Graded Risk Assignment. Note that six categories of risk are listed.
- Use the Recommended Practices Table 1.2, Graded Approach to Software Quality Engineering Recommended Practices V1.1 to provide guidance on the formality for implementing the various recommended practices.

Step 2. Assessing Existing Software Practices (Section 1.3)

The Recommended Practices Checklist for Code Project Practices Assessment (Appendix B) is a tool for understanding the existing software practices. It is recommended that a software quality engineer perform the assessment:

- Periodically to identify the current state of project practices
- To indicate areas for improvement

Step 3. Mapping Results to the Graded Approach Recommendations

The Checklist contains questions specific for formal and informal implementation of a practice. The responses to these questions will assist in the mapping of current practices during the project risk level assessment.

Example:

- A project has been assessed at mid-risk.
- The project answered, "No" to the Peer Review question (formal implementation) "Are reviewers trained to perform their roles?" in the code practices section of the checklist.
- Recommended Practices Table 1.2 indicates that for a mid-risk project, peer review practices most likely should be formal.

In this example, as in practice, both the risk level assessment and the existing practice question's response should be revisited to confirm both are accurate. If found to be accurate, then as in this example, the Peer Review practice may be a candidate for software process improvement since the existing practices may not match the recommended practices for that particular risk level.

Step 4. Assessing and Planning SPI Activities (Section 1.3)

When considering practices to be implemented or improved, a project should weigh these seven factors:

- Type of software
- Project size

- Resources
- Difficulty
- Impact
- Long-term benefits
- Appropriate fit within the life cycle of the project. For example, full software configuration management and tracking of defects would probably be counterproductive early in the code implementation phase.

The software process improvement activities are projects and should be managed appropriately

- Define the scope
- Understand the objectives
- Identify the resources, money, and manpower
- Note obstacles
- Discuss contingencies
- Generate an action plan

Refer to the Recommended Practices Figure 1.2

Step 5. Implementing Software Process Improvement Action Plan (Section 1.3)

Once a decision has been made to change and improve a process:

- The project leader uses discretion in prioritizing, planning, scheduling, and deploying specific improvements.
- The small working group creates the action plan, using the template available at http://www.llnl.gov/asci/sqe/
  —to detail the level of formality of the activities and work products of the project
  —to explore the options for change
  —to accomplish the recommended software process improvement
  —to detail the transition from the current state to the recommended state
- The working group typically gains a thorough understanding of the current practice to be improved
- Current project goals and objectives drive the improvement
- As a working document, the Action Plan will likely change as the working group further defines and implements the process improvement
- With the project leader's approval, the working group can try, evaluate, and revise a pilot
- If the pilot is successful and the team is agreeable, the working group can install the practice and train all project participants
- A trial period of a few months will allow issues to surface around the new (or revised) practice, and further improvements may be made

Guiding Principles and Management Practices (Section 2.)

- The four guiding principles and management practices in this section are broad in scope, and are grouped together.
  - —people are the most important software quality resource
  - —interorganizational coordination
  - —training
  - —using metrics to manage

GPM Practice 1. People Are the Most Important Software Quality Resource

People are the most important SQE resource. They furnish the time, energy, and intellectual effort that contribute overwhelmingly to project success in the LLNL research and development environment. When an organization treats its people as valued resources, it can

- Retain excellent people,
- Attract and hire new excellent people,
- Make better decisions about technical directions and choices, and
- Provide more innovative and outstanding products.

GPM Practice 2. Interorganizational Coordination

Interorganizational coordination facilities cooperation and reduces redundancy among the various organizations and teams working on a project involving software engineering.

- Software engineers work with customers and end-users to establish system requirements
- Plans and activities are coordinated with all affected groups
- System requirements and project objectives are reviewed by representatives of all those affected
- An environment is created facilitating interaction, coordination, support and teamwork
- Work products
  - —written plan for communicating interorganizational commitments, coordination and tracking of work
  - —documentation of V&V processes used to certify correctness of software product
  - —documented procedure to resolve interorganizational issues
  - —technical review work products
  - —test plans and testing relating work products

GPM Practice 3. Training

Effective training develops the skills and knowledge of project members so that they can perform their job assignments productively and efficiently. The most effective training is planned and integrated with other project practices and activities.

- Evaluate current and future skill needs
- Identify the training needed by the project team and individual members
- Procure or develop training to address the training needed
- Work products
  - —project training plan
  - —project training records
  - —training materials
  - —training schedule

GPM Practice 4. Using Metrics to Manage

Metrics and software measurement can provide information that helps in the planning, execution and control, and review and evaluation of a project. Measurement can help assess the benefits (in terms of productivity and quality) derived from new software engineering methods and tools.

- Plan management use of metrics to make decisions based on fact
- Select appropriate measurements based on specific project management objectives
- Take measurements
- The way in which documentation is used to record metrics data and management decisions based on it depends on what measurements are being made.
- Utilize metric thresholds
- Work products
  - —metrics sections in planning documents, such as the SDP

Software Verification Recommended Practices (Section 3.2)

Software verification determines the following:

- The requirements are accurately and correctly implemented
- That such requirements are adequate, given the intended uses of the software

The single practice, Testing and Other Software Verification Activities, is closely integrated with many others in the Software Engineering Recommended Practices.

SV Practice 1. Testing and Other Software Verification Activities Practice

This practice includes the following:

- Various software testing activities
  - —unit
  - —integration
  - —system
  - —acceptance testing
- Using tools to assess the code coverage during testing activities

- Performing a series of tests, generally system level tests
  —after each nightly build
  —prior to code being placed in a baseline repository
- Test result verification techniques to compare actual results with expected results
  —human analysis based on intuition or
  —the development of support verification software
- Use of software test tools to perform automated testing for the various testing methods
  —may be commercial or in-house tools
  —memory leak detection tools
- Activities to verify compliance to applicable coding and language standards
  —naming conventions
  —robustness design
  —style of comments
  —resolution to compiler warnings
- Work products
  —software verification and validation plans
  —software test plans
  —software test cases and procedures for unit or module testing
  —integration testing
  —system level testing
  —acceptance testing

Software Engineering Recommended Practices (Section 3.3)

Software engineering is defined as the systematic, disciplined, and quantifiable approach to the development, operation, and support of software (i.e., the application of engineering to software).

- The six practices in this area are as follows:
  —software product engineering
  —interface definition and control
  —data and database interoperability assurance
  —peer reviews
  —software configuration management (SCM)
  —software quality assurance (SQA)

Some practices, however, may also affect activities in software verification or software management areas.

SE Practice 1. Software Product Engineering Practice

This practice treats a software project as a disciplined engineering process with well-defined activities.

- Project members
  —perform these activities
  —monitor and improve their performance
  —hold themselves accountable for quality control
- The Software Product Engineering practice spans activities in every aspect of the software product life cycle, "from cradle to grave." Other life cycle activities performed such as verification and validation, including software testing, are described in the Testing and Other Verification Activities Practice. Software Product Engineering includes
  —project planning
  —product definition and specifications
  —product design, implementation and test
  —product release
  —product operations and maintenance
- Tailoring
  —no two projects have identical characteristics and constraints
  —it is unlikely that any two project-specific Software Development Plans (SDPs) will call for exactly the same product engineering activities
  —most projects require a subset of the process activities and deliverables listed
  —each project should tailor this process by preparing project-specific plans that map these practices to fit the applicable constraints and realities
- Work products
  —Product Requirements Description
  —System Requirements Description
  —Software Requirements Specification
  —Software Development Plan
  —Software Quality Assurance Plan
  —Software Configuration Management Plan
  —Software Design Description

SE Practice 2. Defining and Controlling Interfaces Practice

This practice defines and controls interfaces by developing a complete description of the arguments and tracking interfaces to minimize unexpected changes.

- The practice generally contains sequential activities
  —starting with building a skeleton of the interfaces
  —adding basic functionality such as returning dummy values to "talk to" other components until a complete end-to-end structure is built
  —completing functionality when the skeleton correctly executes all its required functions with dummy return values and stubbed-out subroutines
- Good implementation practices include grouping variables into one or more data structures to avoid problems that may develop when a variable is propagated through a very long chain of subroutines

- Work products
  - —initially component or module stubs
  - —later evolving to fully functioning components and modules

SE Practice 3. Ensuring Data and Database Interoperability Practice

This practice standardizes the format and interpretation of data transferred between multiple application systems.

- As interoperability factors change, earlier decisions should be revisited
- This practice produces data files and databases that
  - —meet the appropriate development standards and methodologies
  - —reduce data redundancy
  - —maintain data and referential integrity
- Work products
  - —data and databases design documentation.

SE Practice 4. Peer Reviews Practice

In this practice, an author's peers methodically examine his or her work products to identify defects and areas where changes are needed.

- Peer reviews vary in level of formality, from formal inspections to informal walkthroughs.
- Each project should adapt the peer review process that meets its needs
- The following should be considered when tailoring the review process
  - —the number of products subject to peer review
  - —team member viewpoints
  - —checklists specific to project requirements
  - —metrics to be collected
- Work products
  - —peer review meeting report
  - —peer review anomaly detail and summary

SE Practice 5. Software Configuration Management Practice

The software configuration management (SCM) practice establishes and preserves product integrity throughout the life cycle of the software.

- SCM provides the following practices:
  - —identify the configuration of the software at given points in time
  - —systematically control changes to the configuration
  - —preserve product integrity and traceability throughout the life cycle

- Appropriate SCM activities and work products should be graded based on the following aspects of the project:
    —maturity
    —size
    —importance
    — risk
- Work products
    —reports from code management tool
    —defect and problem reports
    —enhancement requests
    —a software configuration management plan
    —CCB meeting minutes
    —CCB change request evaluation notes

SE Practice 6. Software Quality Assurance Practice

The software quality assurance (SQA) process

- Improves the quality of software products,
- Strengthens customer satisfaction,
- Provides managers with appropriate visibility into the project processes and products.
- Includes verification activities
    —test
    —reviews
    —traceability methods
- Assures that a project is following its agreed practices and processes
- In large, high- and mid-risk projects, a specialized dedicated team more easily performs SQA activities
- To ensure compliance to applicable governing documents and plans, the SQA team participates in the following:
    —preparation, selection, and review of any project development plans, processes, and standards
    —reviews of planned project software engineering activities and software products
    —collection of measurements and generation of metrics to evaluate the quality of the product
    —assessment of the value added when conducting software process improvement activities
- Work products
    —documented plans, processes, and standard
    —results of reviews, and metrics analysis
- Associated work products from the Software Project Management practice include
    —software quality assurance plan
    —software configuration management plan

Software Project Management Recommended Practices (Section 3.4)

Software project management is defined as the systematic approach to balancing project work, required resources, methodologies, procedures, schedules, and project organization.

- Software project management includes the following three practices:
  - —project planning, tracking, and oversight
  - —requirement management
  - —risk management

Some practices may also affect activities in software engineering or software verification

SPM Practice 1. Project Planning, Tracking, and Oversight Practice

Project planning, tracking, and oversight practices are applied across the entire software life cycle, and thus include both development and operation.

- Project planning is selectively applied and tailored to fit a particular project
  - —identifies tasks, activities and practices required for achieving project goals
  - —defines a plan to perform the work
  - —establishes commitments among participants
  - —allocates, organizes, and schedules equipment and personnel resources to complete the software product on time and within budget
- Software planning begins with a statement of the work to be performed and other constraints and goals that define and bound the software project
  - —estimate sizes of software work products and resources needed
  - —produce a schedule, identify
  - —assess software risks
  - —negotiate commitments
- Tracking and oversight provide visibility into actual project progress
  - —managers can take effective action when project performance deviates considerably from defined project planning
  - —a documented plan (SDP) tracks software activities, communicating status and revising plans
- Work products
  - —a tailored software development plan or simple project plan
  - —a project schedule
  - —project review reports
  - —a software quality assurance plan
  - —a software configuration management plan

SPM Practice 2. Requirements Management Practice

The requirements management practice establishes and maintains a common understanding between the customer and the project regarding the software requirements.

- This agreement forms the basis for estimating, planning, performing, and tracking project activities throughout the software life cycle
- The project gathers requirements from a number of sources
  - —interviews with potential users
  - —reviews of competitive products
  - —interactive prototypes and test problems
- Changes to the requirements should be reviewed, accepted, and incorporated into the software project in a timely fashion
- Modifications should be controlled with a formal or informal change-management (CM) process, depending on the project
- Software tools are available to assist in the management of requirements
- Work products may be produced from the activities in this practice
  - — documentation (usually in an SRS) of clearly stated, consistent and testable requirements
  - —minutes of review meetings
  - —lists of incomplete and missing requirements

SPM Practice 3. Risk Management Practice

By implementing risk management techniques, projects can reduce the risk of negative impacts to a more acceptable level.

- Risk management practices help to identify, analyze and control project areas or events (problems) with the potential to detrimentally affect the project
  - —schedule delays
  - —increased costs
  - —low quality
- A risk management plan (RMP) can be written to accomplish the following:
  - —identify risks
  - —assign responsibilities for management or mitigation
  - —track the results of risk-limitation activities
- Risk assessment and risk-control measures address the nature and potential impacts of likely risks
  - —risks are analyzed and assessed
  - —the project should apply an appropriate risk-control strategy
  - —risk-control measures may be graded to suit individual projects
- Work products
  - — risk-management plan